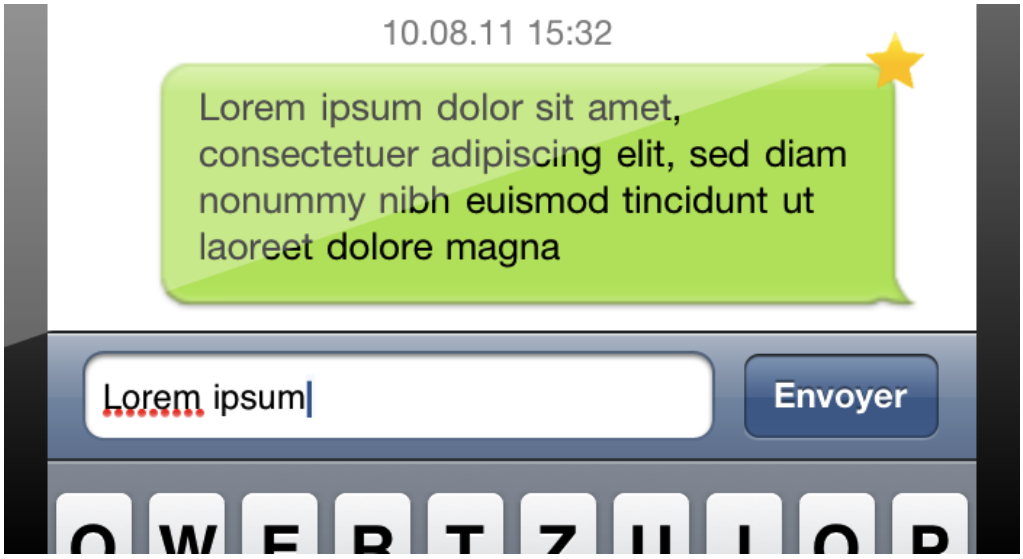
Hes·so VALAIS WALLIS
Haute Ecole Spécialisée de Suisse occidentale
Fachhochschule Westschweiz
University of Applied Sciences Western Switzerland

Travail de Bachelor 2011

Une application chat dans le système iOS



Etudiant : Daniel Hunacek

Professeur : Nicole Glassey

www.hevs.ch

❖ Résumé

Ce travail de Bachelor a pour objectif de proposer, dans le cadre du projet MMHS (My Mobile Health Space), une application iPhone de messagerie instantanée. Celle-ci est tout d'abord destinée à de jeunes asthmatiques âgés de huit à dix-huit ans, puis par la suite, à n'importe quel type de patient atteint d'une maladie chronique.

Cette application est créée dans le but d'être intégrée par après dans une application plus globale du projet MMHS. Cette dernière est censée mettre à disposition de patients des outils leur permettant d'appréhender leur maladie de manière positive.

Pour développer cette application de messagerie instantanée il a fallu établir, dans un premier temps, les besoins des utilisateurs, puis dans un second temps, proposer une solution logicielle composée de plusieurs fonctionnalités.

Proposer une application mobile de ce genre nécessite une certaine architecture. Il faut permettre à deux personnes de pouvoir communiquer via leur Smartphone. La solution proposée comporte trois parties :

- Une application mobile sur iPhone
- Un service web
- Un service de notification

L'application se connecte au service web pour envoyer et recevoir des messages. Elle permet en outre de rechercher sur le service web des contacts afin d'ajouter des amis à sa liste de contacts. Enfin, elle propose à l'utilisateur de pouvoir marquer certains messages comme favoris, pour les retrouver facilement.

Le service web est un serveur PHP avec base de données en MySQL qui met à disposition une liste de fonctionnalités, utilisables via un Smartphone.

Le service de notification quant à lui est chargé d'envoyer une notification Push sur l'iPhone d'un utilisateur. Pour ce faire, il va contacter un service proposé par Apple nommé APNs (Apple Push Notification Service) et c'est l'APNs qui va ensuite être capable de contacter l'appareil en question. Ce service est sur le même serveur que le service web. Il est aussi développé en PHP et utilise la même base de données.

❖ Tables des matières

1 Introduction 1

2 Présentation générale 2

3 Analyse des besoins 3

3.1 Les amis 3

3.2 Les discussions 3

3.3 Les favoris 3

4 Architecture 4

5 Application cliente 6

5.1 Fonctionnement 6

5.1.1 Login 6

5.1.2 Amis 6

5.1.3 Discussions 7

5.1.4 Favoris 7

5.1.5 Paramètres 7

5.2 Ecrans 8

5.3 Architecture 14

5.3.1 Modèles 14

5.3.2 Vues 16

5.3.3 Contrôleurs 16

6 Service web 17

6.1 Présentation 17

6.2 Technologies utilisées 17

6.3 Fonctionnement 17

6.3.1 Format d'échange 18

6.3.2 Librairie 18

6.4 Architecture 19

6.4.1 Sessions 19

6.4.2 Arborescence du service web 20

6.4.3 Modèle de données 21

7 Service de notification 23
7.1 Présentation 23
7.2 Fonctionnement 23
7.2.1 APNs 23
7.2.2 Contenu d'une notification 26
7.3 Mise en place du service 28
7.3.1 Choix du service de notifications 28
7.3.2 Génération du certificat 28

8 Gestion de projet 29

9 Bilan 31
9.1 Difficultés rencontrées 31
9.1.1 Application Chat 31
9.1.2 Service web 31
9.1.3 Service de notification 32
9.2 Améliorations possibles 32
9.2.1 Application Chat 32
9.2.2 Service web 33
9.3 Bilan personnel 33

10 Conclusion 34
10.1 Mot de la fin 35

11 Remerciements 36

12 Déclaration sur l'honneur 37

13 Sources 38
13.1 Bibliographie 38
13.2 Webographie 38
13.2.1 Développement iPhone 38
13.2.2 Architecture 39
13.2.3 Service web 39
13.2.4 Service de notification Push 39

14 Glossaire 40

15 Table des illustrations 41

15.1 Images....41
15.2 Tableaux....42
15.3 Exemple de code....42

16 Annexes....43

1 Introduction

Ce travail de Bachelor s'affilie au projet MMHS (My Mobile Health Space) de l'institut d'informatique de gestion de la HES-SO de Sierre. MMHS est un projet Ra&D (Recherche appliquée et développement) consistant à développer des services mobiles innovants destinés aux personnes atteintes de maladie chronique. L'objectif étant qu'au travers d'applications ludiques, le patient puisse appréhender de manière positive sa maladie.

Le projet MMHS prévoit des services dans les trois thèmes suivants :

- le jeu formatif
- le suivi médical
- la communauté

C'est dans ce dernier point que ce travail de Bachelor vient s'inscrire. Il va permettre au patient de pouvoir être mis en relation avec d'autres personnes dans la même situation, afin d'obtenir un suivi et des conseils à propos de sa maladie.

L'objectif de ce travail est de créer une application mobile de messagerie instantanée permettant à plusieurs personnes de se connecter et de pouvoir discuter de leur maladie. Ils peuvent ainsi mettre à contribution des autres leur expérience personnelle.

L'ère du mobile a rencontré un tournant important durant les dernières années avec l'arrivée du Smartphone. Ce dernier est en plein essor et les systèmes pouvant y être embarqués sont multiples. Cependant deux d'entre eux prédominent le marché : Android développé par Google et iOS développé par Apple. C'est sur la plateforme d'Apple, plus précisément sur iPhone, que cette application est développée, ceci en exploitant les notifications Push afin de rendre l'interaction entre les utilisateurs quasiment instantanée.

2 Présentation générale

Le projet s'adresse principalement à un public jeune, c'est-à-dire à des enfants âgés de 8 à 12 ans et des adolescents ou jeunes adultes âgés de 13 à 18 ans. C'est sur les enfants de l'école de l'asthme à Sion que le projet MMHS a décidé de se concentrer dans un premier temps, afin d'apprendre à ces jeunes de manière ludique à vivre avec leur asthme.

Cette application mobile est développée sur le système iOS d'Apple dans sa dernière version stable actuellement, c'est-à-dire iOS 4.3 et exploite le système de notifications Pushd'Apple afin d'avertir l'utilisateur d'un nouvel événement.

En résumé, le but de ce travail est de mettre en place une application de chat sur iPhone afin que ces jeunes puissent discuter entre eux à propos de leur asthme. Cette application de messagerie instantanée viendra ensuite se greffer au projet MMHS.

Cependant en parlant de chat sur iPhone, notre première réaction est : « Mais une application de chat sur iPhone ça existe déjà ! ». Certes, de nombreuses solutions existent sur iPhone. La plus connue est sans doute Whatsapp, une application très complète permettant de converser avec des amis. Elle est disponible sur iPhone et sur Android, ce qui a contribué à son succès. Elle permet en outre d'envoyer des images et des notes audio.

Une particularité de Whatsapp est qu'elle fonctionne avec le numéro de téléphone de l'utilisateur. Nul besoin de créer un compte manuellement avec mot de passe, question secrète et email valide. Tout est fait de manière transparente et en utilisant le numéro de téléphone comme identifiant. Malheureusement, en exploitant cette technique, l'usage de Whatsapp est restreint aux téléphones mobiles uniquement. Il n'est donc pas possible d'utiliser Whatsapp sur un iPod Touch, iPad ou toute autre tablette mobile.

Une telle utilisation n'est pas envisageable dans ce travail, étant donné que cette application est censée venir s'ajouter à une plus grande application. Il faut que l'utilisateur soit enregistré et qu'il ait un compte en ligne, qui lui permettra d'accéder à toutes les ressources de l'application.

D'autres applications existent bien évidemment et n'utilisent pas le numéro de téléphone comme identifiant. Mais celles-ci sont bien souvent liées à un ou plusieurs comptes de messagerie instantanée tels que gTalk, msn messenger, skype etc. Il n'est donc pas non plus concevable d'utiliser un service de ce genre pour venir s'intégrer au projet MMHS.

Il faut revoir cette application depuis le départ, en prenant en considération les besoins du projet MMHS.

3 Analyse des besoins

L'objectif est de fournir au patient un moyen de communiquer avec les autres internautes et de partager diverses informations. L'application doit offrir à l'utilisateur la possibilité de se créer une liste d'amis et de converser avec ces derniers.

Elle doit en outre permettre de sélectionner certains messages jugés utiles afin d'être enregistrés comme favoris et facilement récupérables.

De ces objectifs résulte une liste de fonctionnalités prioritaires et secondaires définies en début de projet. Ces dernières sont décrites dans le cahier des charges disponible en annexe.

3.1 Les amis

L'utilisateur doit pouvoir chercher un contact dans la liste des personnes utilisant ce chat et effectuer une demande d'ajout. Cette demande doit ensuite être validée par l'utilisateur en question pour que le lien d'amitié soit effectif.

3.2 Les discussions

Chaque utilisateur doit ensuite pouvoir sélectionner un contact de sa liste d'amis et commencer une conversation en lui envoyant un message. La conversation comprendra tous les messages que les deux intervenants se seront envoyés. Chacun sera libre ensuite de gérer comme il l'entend, le contenu de la conversation affichée sur son application. Il pourra supprimer des messages voire la conversation complète.

3.3 Les favoris

Les conversations qu'ont les interlocuteurs sont censées être principalement en rapport avec leur maladie et certains messages peuvent être des avis constructifs et utiles. Ces astuces doivent ensuite être rapidement retrouvées par l'utilisateur. Afin de lui éviter de rechercher dans toutes les conversations qu'il a eues, une page contenant la liste des favoris est disponible.

4 Architecture

De prime abord la communication entre deux Smartphones peut paraître triviale. Cependant elle ne consiste pas seulement à communiquer d'un appareil à un autre via une connexion directe comme le Bluetooth ou l'infrarouge. Etant donné que l'utilisateur doit pouvoir avoir accès à son service depuis n'importe quel endroit, la communication ne peut se contenter d'utiliser un des systèmes précédemment mentionnés qui oblige l'utilisateur à se trouver à proximité de son interlocuteur. Il faut donc passer par Internet afin de pouvoir envoyer un message sans se soucier de la localisation du destinataire.

De plus, le destinataire n'est pas forcément en ligne au moment de la réception du message. Le message doit donc être stocké jusqu'au moment où la personne se connectera, un peu comme pour un e-mail.

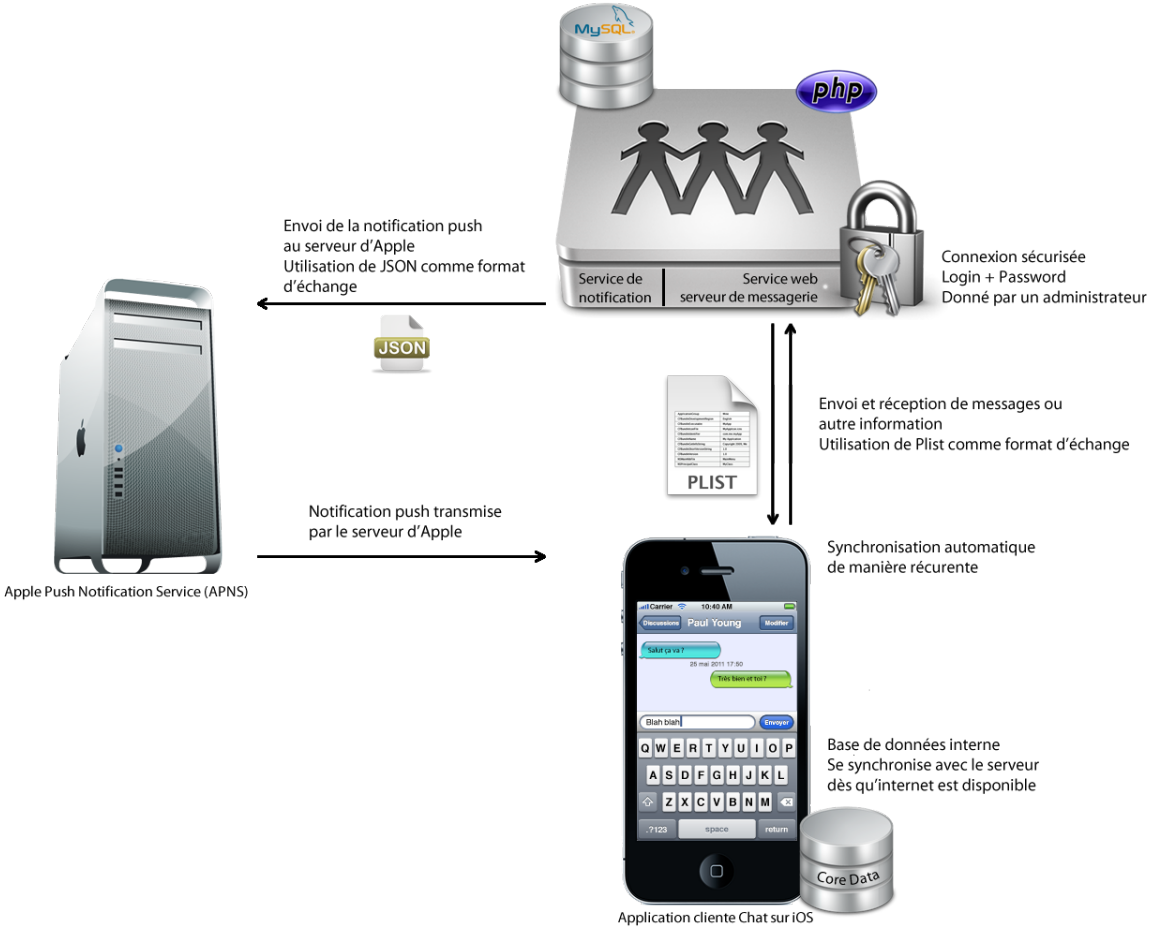
Afin de pouvoir répondre à ces critères, il faut adopter une architecture client – serveur, autrement dit créer un service web qui fera office de relai, agissant comme un facteur délivrant son courrier. Il met les messages en attente dans une boîte aux lettres et lorsque la personne se connecte, elle va récupérer son courrier.

Après se pose un autre problème. En effet nous voulons que les communications soient instantanées. Il faut donc avertir le destinataire de la réception d'un nouveau message. C'est là qu'intervient le système de notifications Push, qui va contacter la personne concernée en lui envoyant une alerte sur son téléphone, même si celui-ci est en veille.

Nous avons donc trois parties dans notre architecture : l'application cliente sur le Smartphone qui est chargée d'envoyer et récupérer les messages, le service web qui lui, stocke les messages et le service de notification qui averti l'utilisateur de la réception d'un message.

Ci dessous se trouve un schéma de l'architecture relative à ce projet. Il comporte certains éléments et choix techniques qui seront décrits plus loin dans ce rapport. Cependant nous pouvons déjà placer les trois acteurs de cette architecture. Tout d'abord le Smartphone en bas du schéma est l'appareil sur lequel l'application cliente est installée. Ensuite, cette application communique avec le service web, qui se trouve sur le serveur en haut du schéma. Celui-ci est utilisé pour envoyer et recevoir des messages ou tout autre information. Enfin, le service de notification est également installé sur le serveur et va tourner en parallèle du service web. Ce service est chargé d'envoyer les notifications, en contactant le serveur d'Apple APNs (Apple Push Notification Service) pour notifier l'appareil concerné.

Image 1 : Schéma d'architecture



5 Application cliente

L'application cliente est l'élément le plus important pour l'utilisateur, car c'est son seul moyen d'interagir. C'est l'application qui lui permettra de se connecter et de lire et envoyer des messages.

5.1 Fonctionnement

L'application fonctionne comme suit.

5.1.1 Login

A sa première connexion, l'utilisateur est invité à se connecter avec son adresse email et son mot de passe. S'il réussi, il reçoit une clé de session qu'il utilisera après de manière transparente lors de chaque interaction avec le serveur. Si les informations ne correspondent à aucun utilisateur enregistré sur le service web, un message d'alerte l'avertira de cette erreur.

> *Nous partons ici du principe que les utilisateurs existent déjà et qu'ils peuvent être gérés depuis la plateforme phpmyadmin du service web. Puisque cette application est destinée à être intégrée dans une application de plus grande envergure, c'est au moment où cette dernière sera créée que la gestion des utilisateurs devra être mise en place.*

Une fois connectée avec succès, l'application se synchronise avec le service web et récupère les messages, les amis et les favoris de l'utilisateur afin de remplir sa propre base de données.

5.1.2 Amis

La première page à être chargée est la liste d'amis. Sur cette liste apparait la liste des contacts acceptés et en attente d'être acceptés par l'utilisateur connecté. Pour accepter un contact, il suffit de cliquer sur le bouton bleu « accepter » à coté dudit contact. Pour supprimer n'importe quel contact, le bouton « modifier » en haut à gauche de la page permet d'activer le mode édition et de supprimer une entrée de la liste d'amis.

Dans le coin supérieur droit de la page se trouve un bouton « + » qui permet d'ajouter un ami. Ce bouton va ouvrir une page avec un champ recherche permettant de rechercher un ami parmi tous les utilisateurs enregistrés sur la base de données du service web. Le résultat retourné par le service web sera affiché dans une liste au dessous. Et lorsque l'utilisateur clique sur un des contacts proposés dans cette liste, une demande d'ajout sera immédiatement envoyée à cette personne et l'utilisateur sera renvoyé à la page précédente.

Enfin, en cliquant sur un des contacts de la liste d'amis, cela va créer une nouvelle discussion avec ce contact ou rouvrir la conversation précédente si elle existait déjà.

5.1.3 Discussions

La page de discussions affiche la liste des discussions ouvertes par ordre chronologique décroissant. Chaque ligne affiche : le contact rattaché à cette discussion et les deux premières lignes du dernier message reçu ou envoyé. Si un message n'a pas encore été lu dans une discussion, une petite enveloppe apparaît à gauche, signalant le message comme non lu.

En cliquant sur l'une des discussions, cela charge la page de cette conversation. Tous les messages sont affichés par ordre chronologique dans des bulles de couleurs différentes. Une bulle grise à gauche pour un message reçu et une bulle verte à droite pour un message envoyé. En laissant le doigt appuyé sur un message, cela fait apparaître un menu contextuel permettant de choisir entre trois options : copier le texte de la bulle, supprimer le message ou alors placer le message dans les favoris. En choisissant de mettre dans les favoris, une étoile apparaîtra sur le message. Et il sera disponible sur la page des favoris.

5.1.4 Favoris

La page des favoris affiche la liste des messages tagués comme favoris. Il est possible de supprimer n'importe quel message de cette liste en cliquant sur le bouton modifier en haut à gauche. En sélectionnant un message, cela place le contenu du message dans le presse-papier.

5.1.5 Paramètres

La page « paramètres » permet d'ajouter tous les éléments nécessaires à la gestion des paramètres de l'application. L'application n'ayant pas d'options pour le moment, il n'a qu'un bouton « déconnecter » permettant de mettre fin à la session.

5.2 Ecrans

Voici à présent les différents écrans de l'application de chat. Nous allons les détailler un à un afin de montrer comment utiliser cette application.

Image 2 : Ecran de login

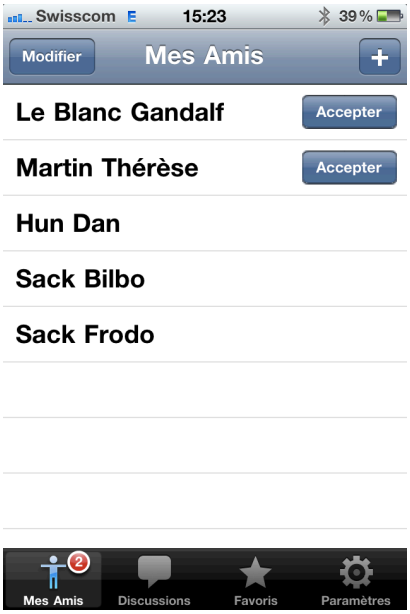
Lors de la première connexion, la page de login apparaît. Les informations à fournir sont l'identifiant (l'email) et le mot de passe.

En cliquant sur le bouton « se connecter », cela va envoyer la requête de connexion.

En cas d'erreur un message s'affichera.

Si tout est en ordre, le processus continuera sur la page « Mes Amis ».

Image 3 : Ecran "Mes Amis"



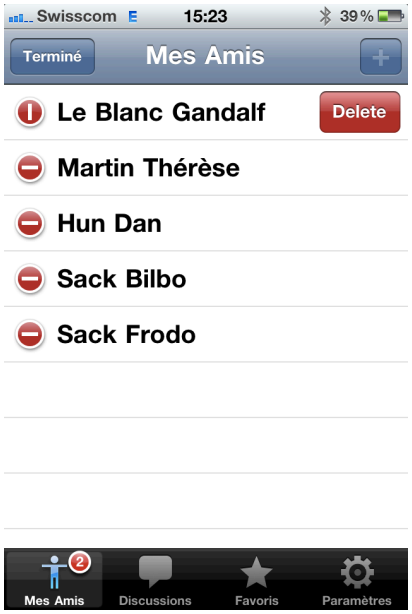
Ceci est l'affichage principal de l'application. La barre noire en bas permet de naviguer entre les différents onglets.

Sur cette page se trouve la liste des amis. Les nouveaux amis sont à accepter avec le bouton à droite du nouveau contact, ou refuser en supprimant le contact (bouton « modifier »).

Pour envoyer un message, il suffit de sélectionner un contact.

Pour ajouter un contact il faut cliquer sur le bouton « + ».

Image 4 : Ecran d'édition de "Mes Amis"

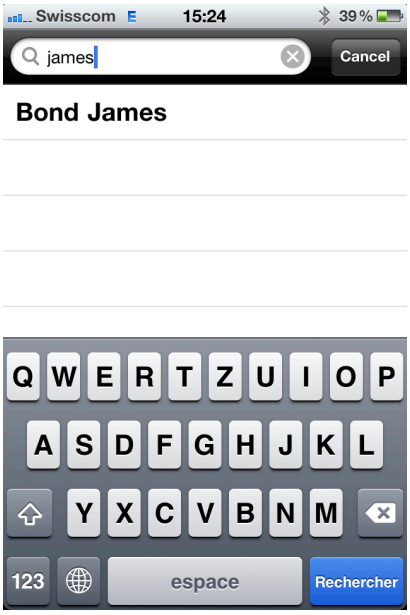


Ceci est la vue lorsque le bouton « modifier » a été cliqué. C'est le mode d'édition de la page permettant de supprimer n'importe quel contact.

Pour supprimer un contact, il suffit de cliquer sur le rond rouge à gauche du contact, puis le bouton « Delete » à droite.

Pour revenir à l'affichage normal, il faut appuyer sur le bouton « Terminé ».

Image 5 : Ecran de recherche d'amis



Ceci est la page de recherche d'amis. Elle survient lorsque le bouton « + », précédemment cité, est cliqué.

Pour rechercher un contact il faut simplement entrer une partie de son nom, puis cliquer sur le bouton « Rechercher » en bas à droite.

Le résultat de la recherche apparaît dans la liste.

Pour demander un ajout d'ami, il suffit de cliquer sur le contact désiré.

Voici le second onglet : « Discussions ». Ici sont listées toutes les conversations ouvertes, classées par contact, avec le dernier message affiché.

Si le message n'a pas encore été lu, une petite enveloppe est affichée à gauche du message.

Les éléments les plus récents sont au début.

Le bouton « modifier » permet de supprimer une conversation.

Le bouton en haut à droite permet de commencer une nouvelle conversation.

Pour ouvrir une conversation, il suffit de cliquer sur un élément de la liste.

Image 6 : Ecran "Discussions"

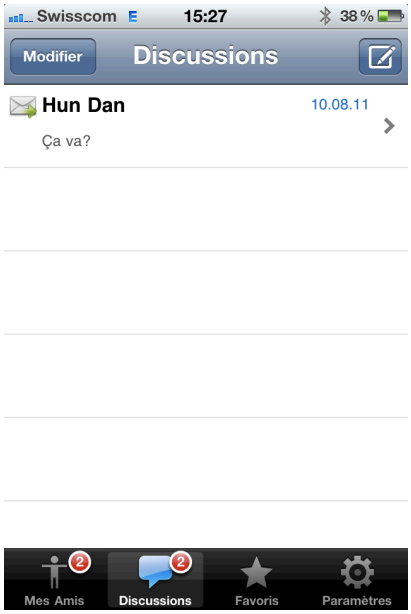
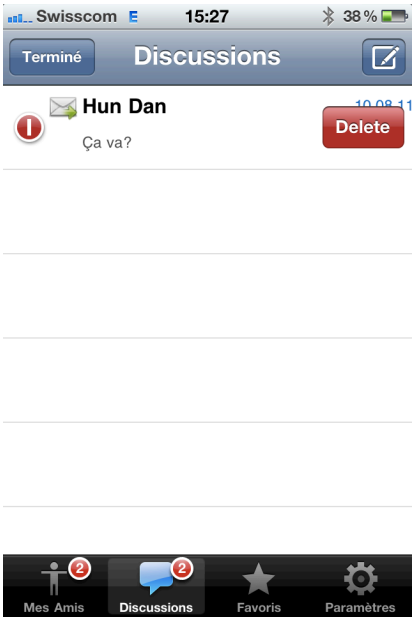


Image 7 : Mode d'édition de "Discussions"



Ceci est le mode d'édition de la page « Discussions ». Il est possible de supprimer un élément en cliquant sur le rond rouge à sa gauche, puis en cliquant sur le bouton « Delete » à droite.

Pour revenir au mode normal, il faut cliquer sur « Terminé ».

Voici la vue d'une conversation, c'est-à-dire la conversation sélectionnée à la page « Discussions ».

Tous les messages avec ce contact sont affichés ici.

En cliquant sur le champ texte en bas, cela ouvre le clavier afin d'écrire du texte. Le bouton « Envoyer » permet ensuite d'effectuer l'envoi du message.

Les messages avec une étoile sont des favoris.

Le bouton « Discussions » en haut à droite permet de revenir à la liste des conversations.

En laissant le doigt appuyé sur un message, cela ouvre un menu contextuel décrit dans la prochaine étape.

Image 8 : Ecran de conversation

Image 9 : Menu contextuel d'une conversation

Voici le menu contextuel lorsqu'on laisse le doigt appuyé pendant un instant sur un message. Il permet soit de copier le contenu du message dans le presse-papier, soit de supprimer le message ou alors d'ajouter le message aux favoris.

En cliquant sur « Favori », une étoile apparaît sur le message pour indiquer que le message est un favori.

Pour supprimer un favori, il suffit de refaire la même manipulation qu'au point précédent, et l'étoile sera enlevée. Cliquer n'importe où permet de faire disparaître le menu contextuel.

Ceci est la page des favoris. On y accède en cliquant sur le bouton « Favoris » dans la barre de menu noire en bas.

Tous les éléments indiqués comme favoris apparaîssent sur cette page classés par ordre chronologique décroissant.

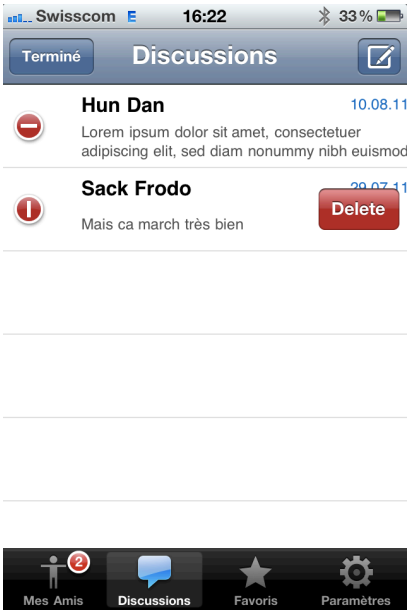
La taille des lignes s'adapte en fonction du contenu du message.

En cliquant sur un élément, le contenu du message est copié dans le presse-papier.

Pour supprimer un message, il faut cliquer sur le bouton « Modifier » afin d'accéder au mode d'édition.

Image 10 : Ecran "Favoris"

Image 11 : Mode d'édition de "Favoris"

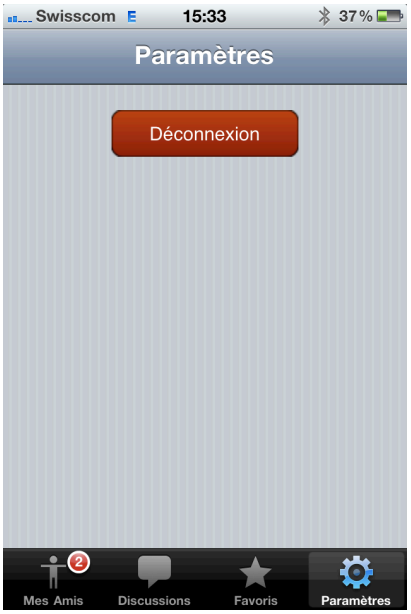


Ceci est le mode d'édition des favoris.

Pour supprimer un élément, il faut cliquer sur le rond rouge à sa gauche et cliquer sur le bouton « Delete » pour valider la suppression.

Le bouton « Terminé » permet de revenir au mode normal.

Image 12 : Ecran "Paramètres"



La page « Paramètres » affiche les différentes options disponibles pour cette application.

Etant donné qu'aucune option n'est disponible pour le moment, cette page ne contient que le bouton permettant de se déconnecter.

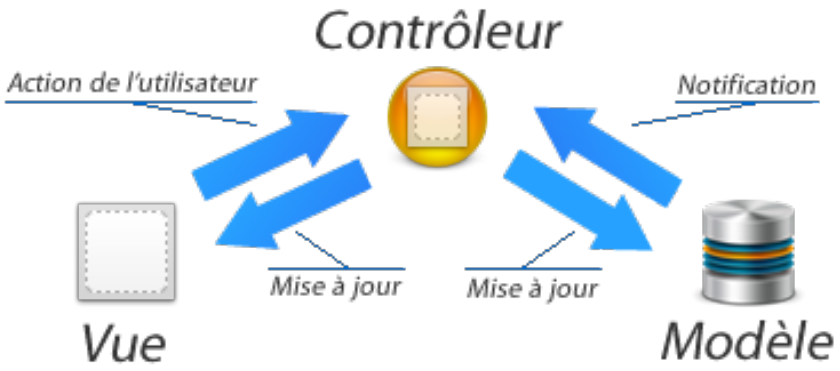
5.3 Architecture

Nous allons à présent passer à une partie plus technique concernant l'architecture de l'application cliente.

Cette architecture est dictée par le SDK (System Development Kit) d'Apple. En effet le SDK suit une architecture MVC (Modèle Vue Contrôleur), c'est-à-dire que les objets de notre application peuvent avoir l'un des trois rôles suivants : modèle, vue ou contrôleur. L'affectation d'un rôle va définir la fonction de cet objet et la façon dont il va communiquer avec les autres objets de l'application.

En bref, les vues sont les objets utilisés pour afficher des éléments à l'écran, alors que les modèles sont la logique de l'application. Ils sont chargés en outre de faire le lien avec la base de données. Les contrôleurs, quant à eux, sont les éléments centraux de cette architecture. Ils vont faire appel à des modèles pour aller chercher des informations dans la base de données, puis ils vont ensuite traiter ces données, pour enfin afficher du contenu à l'utilisateur grâce aux vues. Ces interactions sont illustrées dans le schéma ci-dessous.

Image 13 : Schéma MVC



5.3.1 Modèles

Les modèles sont des objets dont la fonction est d'assurer la logique de l'application. Leur principale tâche est de faire le lien avec la base de données. Ils servent à représenter le contenu de la base de données sous forme d'entités avec

des attributs. Par exemple, un message est une entité qui possède un contenu (le texte du message), un contact, une heure de réception/envoi, etc.

Les modèles ont en général toutes les fonctions leur permettant de se connecter à la base de données. Cependant, ici, nous avons utilisé un framework proposé par Apple qui se charge du stockage et de la récupération des données dans la base de données. Il permet non seulement de créer un objet à partir du contenu de la base mais également de modifier une entrée en synchronisant la base de données avec le contenu d'un objet. Ce framework s'appelle Core Data.

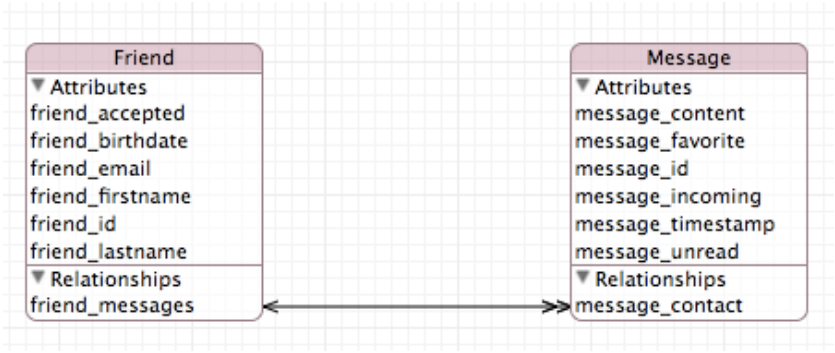
5.3.1.1 « Core Data »

Core Data est le framework utilisé dans cette application afin de gérer les données à persister. Il est chargé de sauver des objets dans la base de données et de les récupérer ensuite afin de les utiliser dans l'application.

5.3.1.2 Modèle de données

Avec Core Data il est très aisé de créer un modèle de persistance. Il y a un éditeur de modèle de données intégré à Xcode qui permet de créer visuellement notre schéma. Le schéma représente toutes les entités qui nécessitent d'être sauvegardées. Chaque entité possède des attributs et éventuellement des relations avec d'autres entités.

Image 14 : Entités dans l'application



Comme nous le voyons sur le modèle ci-dessus, l'application comporte deux entités distinctes « Friend » pour les contacts et « Message » pour les messages reçus ou envoyés. Ces entités possèdent une relation qui va dans les deux sens. Effectivement, chaque message est rattaché à un contact externe : son expéditeur ou destinataire et, à l'opposé, chaque contact est lié à une liste de messages. C'est donc une relation « un à plusieurs » qu'il y a entre ces deux entités.

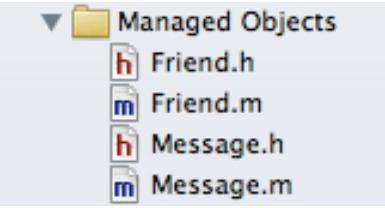
Comparé à celui du service web, que nous verrons plus loin dans ce rapport, ce modèle de données a été simplifié afin de correspondre au mieux avec les besoins de l'application. Inutile donc d'avoir des entités et d'autres relations en plus.

5.3.1.3 « Managed Objects »

Lorsqu'on utilise Core Data, on parle de « Managed Objects », littéralement les objets gérés par Core Data. Chaque objet est un modèle.

Dans l'application iPhone, deux « managed objects » ont été créés, en relation avec les entités générées dans Core Data. Une fois que les objets sont liés aux entités de Core Data, c'est Core Data qui s'occupera de leur instanciation et de la gestion de leurs attributs.

Image 15 : « Managed Objects » dans l'application



5.3.2 Vues

Les vues sont comme leurs noms l'indiquent, les classes permettant de gérer la partie visuelle. Ces classes contiennent toutes les informations nécessaires à l'affichage d'une page et de son contenu. Tous les éléments comme les champs texte, les boutons et images se trouvent dans les vues. Les vues peuvent ensuite être appelées depuis un contrôleur afin d'être affichées sur l'écran de l'iPhone.

5.3.3 Contrôleurs

Les contrôleurs sont les classes responsables de la gestion et de la coordination des vues et des modèles. C'est la logique de contrôle de l'application, qui va gérer les événements. Les contrôleurs vont donc utiliser les objets de la couche modèle pour remplir le contenu des vues et, une fois les vues prêtes, ils affichent les vues. A chaque modification des modèles, les vues sont mises à jour.

Prenons pour exemple la page des favoris. Pour charger la page des favoris, notre application va appeler le contrôleur relatif à la page « favoris ». Une fois le contrôleur instancié, il va récupérer une liste d'objets de type « message » dans la base de données à l'aide de Core Data. Une fois la liste des favoris récupérée, les objets sont utilisés pour remplir chaque ligne de la vue, puis enfin, le contrôleur affiche la vue à l'écran.

Maintenant que nous avons une vision plus précise du fonctionnement de l'application, nous pouvons passer au point suivant qui est le service web.

6 Service web

6.1 Présentation

Le service web est un serveur créé afin de stocker les informations relatives aux utilisateurs : informations personnelles, messages, amis, etc.

Son objectif est de faire le lien entre les différents utilisateurs. Lorsqu'un utilisateur envoie un message à un autre utilisateur, c'est lui qui va faire le relai en stockant le message envoyé dans sa base de données. Il va également fournir une fonction permettant au destinataire de rafraîchir sa liste de messages en téléchargeant les messages sur le serveur. Il en va de même pour sa liste d'amis et sa liste de favoris.

Ce serveur est donc l'élément le plus important, car c'est lui qui va stocker toutes les données. S'il venait à tomber en panne, toute l'application serait compromise. Plus moyen d'envoyer un message ni de les récupérer. Il est donc crucial de mettre le serveur sur une machine fiable et de prévoir un ou plusieurs serveurs de rechange pour prévenir toute panne.

Dans le cadre de ce projet, le serveur est installé sur une machine en DMZ (Demilitarized Zone) de la HES-SO de Sierre. N'étant pas en production, il n'est nullement nécessaire de prévoir un serveur de rechange.

6.2 Technologies utilisées

Le service web est installé sur un serveur web Apache. Le langage de développement utilisé est le PHP et la base de données est en MySQL. Ces choix technologiques ont été faits lors de l'élaboration du cahier des charges, en collaboration avec Guillaume Fort, qui sera chargé de reprendre ce projet par la suite.

6.3 Fonctionnement

Le service web est chargé de traiter toutes les demandes provenant des applications clientes (les utilisateurs). Chaque client doit pouvoir se connecter sur le service web en envoyant des informations et le service web doit lui retourner les données demandées. Par exemple, si le client veut télécharger les nouveaux messages qu'il a reçus, il doit envoyer une requête au serveur et ce dernier va répondre en lui retournant la liste des nouveaux messages.

Pour chaque requête faite par le client, le serveur a besoin d'identifier ce dernier pour des raisons de sécurité. De ce fait, lorsque le client crée sa requête, il doit impérativement fournir en paramètre son id d'utilisateur et sa clé de session reçue lors du login, afin de vérifier qu'il est bien celui qu'il prétend être. Ensuite le serveur va consulter la base de données et retourner les messages au client.

6.3.1 Format d'échange

Cependant, il a fallu définir un format d'échange entre les clients et le serveur. Les plus utilisés sur la plateforme mobile d'Apple sont le format « JSON» et le format « Plist ». Tous deux permettent de transférer des données de manière structurée, mais sur les conseils de Guillaume Fort, notre choix s'est porté sur le format « Plist », un dérivé du format « XML». Ce format étant pris en charge nativement par iOS, il ne nécessite pas l'installation d'une librairie supplémentaire.

6.3.2 Librairie

L'utilisation d'un format d'échange nécessite l'installation d'une librairie capable d'analyser les données à échanger et de générer le fichier qui va être envoyé.

Il n'existe pas énormément de solutions en PHP capable de répondre à ce besoin. Cependant en cherchant un peu sur la toile, une librairie est souvent mentionnée. Il s'agit de CFPropertyList. Elle est très simple à intégrer et elle permet d'une part de lire un fichier « Plist » et en récupérer les valeurs et d'autre part de générer un fichier « Plist » à partir de variables.

Code 1 : Exemple de lecture d'un fichier "Plist" en PHP

```php
// Get the posted parameters
$plistXML = isset($_POST['plistRequest']) ?
      stripcslashes($_POST['plistRequest']) : null;

// Create the plist Request and Response
$plistRequest = new CFPropertyList();
$plistResponse = new CFPropertyList();
$plistResponse->add($rootDict = new CFDictionary());

// Try to parse the posted plist
try {
    $plistRequest->parse($plistXML);
} catch (Exception $e) {
    $rootDict->add('error', new CFString('Unable to parse the
plist.'));
    echo $plistResponse->toXML(true);
    die();
}

// Get the root dictionary
$dictRequest = $plistRequest->getValue()->getValue();

// Get the credentials
$user_id = isset($dictRequest['user_id']) ?
      $dictRequest['user_id']->getValue() : null;
$user_token = isset($dictRequest['user_token']) ?
      $dictRequest['user_token']->getValue() : null;
```

6.4 Architecture

Parlons à présent de l'architecture de ce service web. Il est composé de plusieurs pages web appelées par l'application cliente en fonction de l'opération à effectuer. De ce fait, pour chaque opération que l'utilisateur va effectuer correspondra une page du service web.

Par exemple, si l'utilisateur désire envoyer un message, il va contacter la page « send_message.php » et lui fournir en paramètres les informations relatives au message afin que cette page puisse ensuite faire les opérations nécessaires à l'envoi d'un message.

6.4.1 Sessions

Lorsqu'un utilisateur se connecte au service web, une session est créée. Lors de la connexion, l'utilisateur envoie à la page de login du service web son nom d'utilisateur et son mot de passe haché à l'aide de l'algorithme sha1. Les informations sont ensuite vérifiées et si les informations sont correctes, l'utilisateur reçoit en retour une clé de session, ainsi que sa liste d'amis, ses messages et ses favoris.

Par la suite, lorsque l'utilisateur voudra à nouveau communiquer avec le serveur pour effectuer des opérations, il va devoir fournir en paramètre de chaque requête son id d'utilisateur et sa clé de session. De cette manière, personne d'autre que lui ne pourra effectuer des opérations et une personne ne possédant pas de compte ne pourra pas non plus utiliser le service web.

La session précédemment citée prendra fin au moment où l'utilisateur se déconnectera. Cela aura pour effet de supprimer la clé de session.

6.4.2 Arborescence du service web

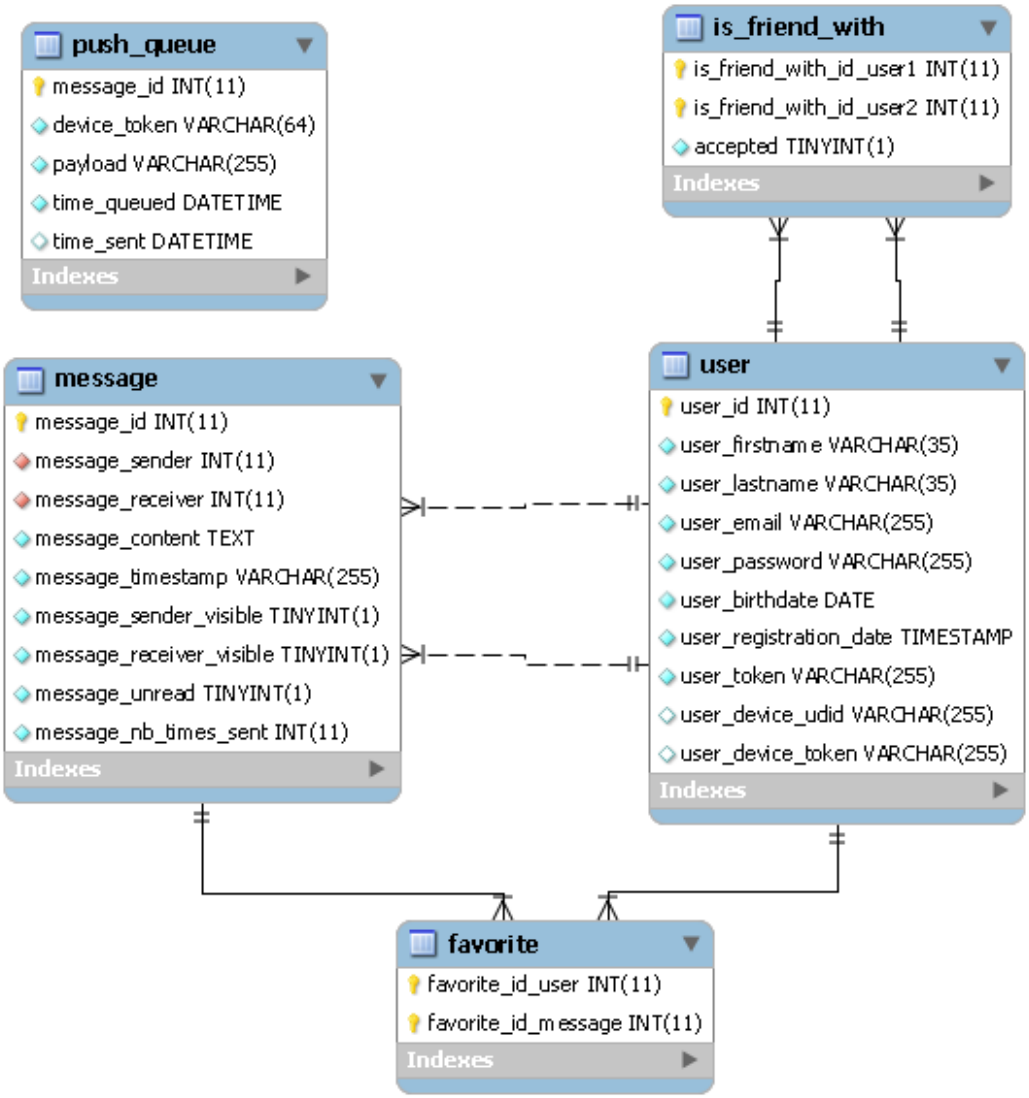
chat-webservice	*Racine du service web*
config	*Fichiers de configuration*
database.php	*Configuration de la base de données*
db_connection.php	*Connexion à la base de données*
lib	*Librairies*
cfproperytlist	*fichiers relatifs à la librairie cfpropertylist*
security	*Fichiers concernant la sécurité*
Session.php	*Classe « Session » permettant la vérification de la validité des informations de sessions d'un utilisateur*
login.php	*Page appelée lors de la connexion* *Paramètres : email et mot de passe* *Retourne : messages, amis et favoris*
disconnect.php	*Fonction : mettre fin à la session*
refresh.php	*Fonction : rafraîchir les amis, messages et favoris* *Retourne : messages, amis et favoris*
search_friend.php	*Fonction : rechercher un utilisateur* *Paramètres : texte de la recherche* *Retourne : liste d'utilisateur*
add_friend.php	*Fonction : ajouter un utilisateur* *Paramètres : id de l'utilisateur*
accept_friend.php	*Fonction : accepter un ami* *Paramètres : id de l'utilisateur*
remove_friend.php	*Fonction : supprimer un ami* *Paramètres : id de l'utilisateur*
update_device_info.php	*Fonction : mettre à jour les infos de l'iPhone* *Paramètres : udid et token de l'appareil*
send_message.php	*Fonction : envoyer un message* *Paramètres : id du destinataire, texte et uuid du message* *Retourne : id, uuid et timestamp du message créé*
remove_messages.php	*Fonction : supprimer une liste de messages* *Paramètres : liste des ids des messages*
update_messages.php	*Fonction : marquer des messages comme lus* *Paramètres : liste des ids des messages*
add_favorites.php	*Fonction : marquer des messages comme favoris* *Paramètres : liste des ids des messages*
remove_favorites.php	*Fonction : enlever des messages des favoris* *Paramètres : liste des ids des messages*

6.4.3 Modèle de données

La base de données relative au service web est sensiblement différente de celle de l'application de chat. En effet elle nécessite de prendre en compte :

- les utilisateurs avec les liens d'amitié qu'ils ont entre eux
- les messages et leurs relations avec les différents utilisateurs
- les favoris liés à un message et un utilisateur.

Image 16 : Schéma de la base de données



Il y a donc deux tables principales dans cette base de données : une table « user » contenant les utilisateurs et une table « message » qui, comme sont nom l'indique, contient les messages. Il y a également une table « favorite » pour les favoris qui n'est autre qu'une table de relations entre un message et un utilisateur. Et pour terminer se trouve la table de relation « is_friend_with », qui permet de définir des liens d'amitiés entre des utilisateurs.

Si nous regardons à nouveau le schéma, nous constatons qu'il y a une table supplémentaire nommée « push_queue ». Cette table n'est pas vraiment en rapport avec le service web. Elle est utilisée par le service de notification. Elle représente la liste d'attente des notifications à faire. Elle ne sera donc lue que par le service de notification. Cependant c'est le service web qui, à chaque nouveau message inséré dans la base de données, est chargé d'ajouter une entrée dans cette table pour dire au service de notification qu'il doit envoyer une notification à l'utilisateur mentionné.

Nous avons vu comment fonctionne le service web et quel est son rôle dans cette architecture ; nous pouvons donc aller à la dernière partie de cette architecture, le service de notification.

7 Service de notification

7.1 Présentation

Le service de notification est responsable d'alerter l'utilisateur sur son Smartphone de tout événement le concernant. En l'occurrence ici ce serveur est chargé d'envoyer une notification à un utilisateur dès qu'il reçoit un message. Ainsi, même si son iPhone est verrouillé, il reçoit une alerte comme quoi un message a été reçu.

7.2 Fonctionnement

Pour envoyer une notification, nous pourrions penser à tort qu'il suffit de contacter directement l'appareil en question, cependant l'envoi de notifications à un appareil sous iOS ne peut être fait en une étape. Apple a tout prévu et fournit un service capable de contacter n'importe quel appareil tournant sous iOS. Ce service, appelé APNs pour Apple Push Notification Service, est un moyen sécurisé d'envoyer un message à un appareil.

Le service de notification de notre projet doit donc utiliser le service d'Apple afin d'envoyer ses notifications au bon destinataire.

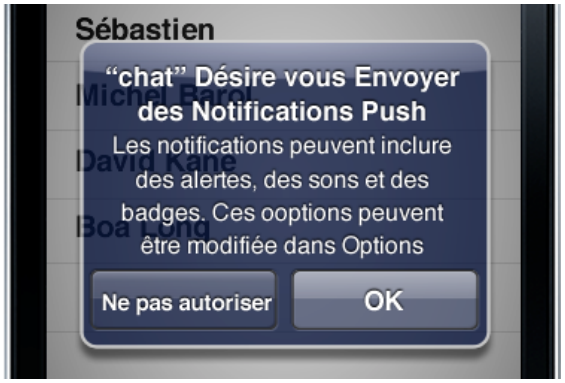
7.2.1 APNs

7.2.1.1 Identification

Afin de contacter un appareil, l'APNs a besoin d'un « token » autrement dit un identifiant unique de l'appareil. Ce token est obtenu automatiquement par l'appareil iOS lorsqu'il souscrit pour les notifications Push auprès de l'APNs. Le token va agir un peu comme un numéro de téléphone qu'on composerait pour envoyer une notification à un appareil.

Avant de pouvoir recevoir des notifications, l'utilisateur devra, au premier lancement de l'application, autoriser cette dernière à lui envoyer des notifications Push. Elle devra ensuite récupérer le token de l'appareil pour le sauvegarder dans les données personnelles de l'utilisateur sur le service web.

Image 17 : Autoriser les notifications Push



7.2.1.2 Certification

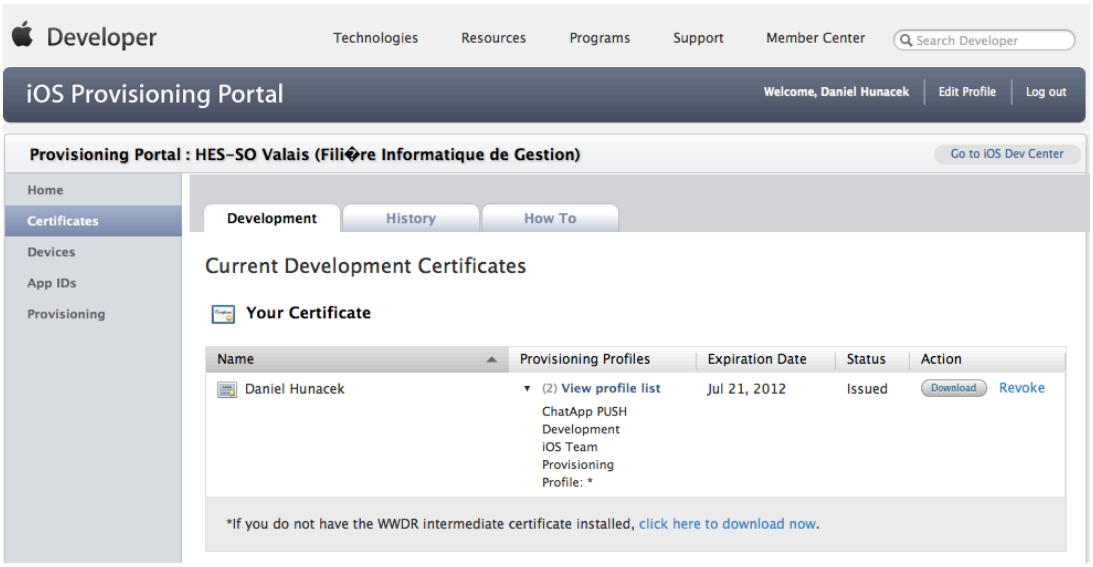
A présent, il faut que notre application puisse être identifiée par le service d'Apple. En effet, en envoyant une requête de notification à l'APNs, celui-ci n'a aucun moyen de savoir quelle application désire envoyer une notification. En matière de sécurité c'est un problème important. Quelqu'un de malfaisant pourrait usurper l'identité d'une application et envoyer des notifications au nom de cette application. Pour pallier à ce problème, l'APNs utilise des certificats afin de permettre de vérifier que l'application qui envoie la notification est bel et bien celle qu'elle prétend.

Le certificat est en quelque sorte la carte d'identité de l'application. A chaque requête qu'il reçoit, le service d'Apple vérifiera l'identité de l'expéditeur en vérifiant son certificat.

Pour générer un certificat à une application, il faut se rendre sur le programme de développement Apple auquel est rattachée l'application. Ce programme est accessible depuis le site web d'Apple dédié aux développeurs à l'adresse suivante :

http://developer.apple.com/

Image 18 : Plateforme des développeurs Apple



https://developer.apple.com/ios/manage/certificates/team/index.action

Ainsi, le certificat est rattaché à une application et lorsqu'une requête est reçue par l'APNs, celui-ci sait quelle est l'application qui essaie d'envoyer la notification. Il peut également faire confiance au certificat, car c'est Apple qui a généré le certificat et que, via des moyens de sécurité liés aux certificats, il n'est plus possible d'usurper une identité en émettant un faux certificat.

Nous n'allons pas aller plus loin dans la certification, car c'est un sujet assez conséquent, sur lequel nous pourrions débattre encore un moment, notamment sur les moyens cryptographiques mis en place pour assurer la sécurité des transactions. Ceci dépasse de loin notre sujet étant donné que c'est Apple qui prend en charge la gestion des certificats. Il est donc inutile de développer plus.

En résumé nous n'avons donc qu'à générer un certificat pour notre application, et le joindre à notre serveur de notification afin que celui-ci puisse contacter l'APNs. C'est ensuite l'APNs qui va faire suivre les notifications aux appareils concernés.

7.2.2 Contenu d'une notification

Dans le cadre de ce projet, il a été demandé d'utiliser les notifications Push comme moyen d'avertir l'utilisateur de l'arrivée d'un nouveau message. Une notification va apparaître sous la forme d'une alerte sur l'appareil iOS. Pour rendre l'expérience de l'utilisateur plus attrayante, l'alerte ne doit pas seulement se contenter de l'avertir au moyen d'un message générique du genre : « Nouveau message reçu ». L'utilisateur désire que le contenu du message reçu soit affiché dans l'alerte. Il n'est ainsi pas forcément obligé de lancer l'application pour lire le message.

Image 19 : Une notification

7.2.2.1 Payload

Pour construire une notification, il faut générer un « payload ». Le payload est un dictionnaire JSON qui va contenir les informations de notre notification. Il ne peut excéder 256 octets, car l'APNs va ignorer tout payload dépassant cette taille.

Le payload se compose comme suit :

Code 2 : Exemple de payload

```
{
    "aps" : {
        "alert" : "You got your emails.",
        "badge" : 9,
        "sound" : "bingbong.aiff"
    },
    "acme1" : "bar",
    "acme2" : 42
}
```

http://developer.apple.com/library/ios/documentation/NetworkingInternet/Conceptual/RemoteNotificationsPG/RemoteNotificationsPG.pdf

Il y a d'abord un premier dictionnaire nommé « aps » qui contient les informations relatives au message d'alerte et à l'application. Tout autre dictionnaire ou variable placé après « aps » est considéré comme paramètre. Ici dans l'exemple « acme1 » et « acme2 » sont des variables passées en paramètres dans notre notification. Elles peuvent ensuite être récupérées dans l'application afin d'être utilisées.

Le dictionnaire « aps » est l'élément le plus important car c'est lui qui contient le message à transmettre à l'appareil. Si l'on prend les éléments les uns après les autres, « aps » peut contenir trois parties :

- « alert » : Une variable de type texte (message à afficher) ou un dictionnaire (*).
- « badge » : Une variable de type numérique qui sera affichée sur le badge de l'icône de l'application.
- « sound » : Une variable de type texte contenant le nom d'un fichier sonore présent dans l'application. Lors de la réception d'une notification, ce son sera joué.

** Dans le premier cas cela signifie que le message d'alerte sera généré en utilisant les paramètres par défaut en affichant le message passé dans la variable texte. Dans le second cas il est possible de personnaliser le message d'alerte en spécifiant différentes options telles que le titre du bouton d'action, la langue (selon la localisation) et l'image à afficher au lancement de l'application.*

Image 20 : Exemple d'un badge

Dans ce projet, c'est le service web qui est chargé de générer le payload à chaque nouveau message envoyé. Il ajoute ensuite une entrée contenant le payload à la table « push_queue » de la base de données du serveur afin que le service de notification récupère ce payload, génère le message et l'envoie à l'APNs. Contrairement à l'exemple ci-dessus, le service web a été configuré pour n'envoyer qu'un message d'alerte contenant le message reçu, précédé du nom du contact, comme le montre l'illustration de notification plus haut (image 19). Le badge et le son n'étant pas utiles pour le moment.

7.3 Mise en place du service

Le service de notification vient en parallèle du service web afin de prendre en charge les notifications. Il a été installé sur la même machine que le service web, étant donné qu'il accède à la même base de données que lui.

7.3.1 Choix du service de notifications

Plusieurs solutions de service de notifications existent déjà en PHP, il était donc plus simple d'adapter le service web afin qu'il puisse cohabiter avec un service de notification plutôt que d'en refaire un. La première solution trouvée était une solution très complète appelé EasyAPNs, et la seconde solution était un exemple de service APNs proposé par Matthijs Hollemans sur un tutoriel complet concernant les notifications, disponible sur le site : www.raywenderlich.com.

La première était peut-être trop évoluée pour envoyer de simples notifications. La seconde, au contraire, était beaucoup plus adaptée. Très simple à configurer et à déployer, elle convenait mieux à notre utilisation. De plus l'adaptation par rapport au service web existant était assez simple.

Il a suffit, dans un premier temps, d'ajouter à la base de données une table « push_queue », agissant comme une file d'attente puis, dans un second temps, de modifier la fonction gérant les envois de messages en y ajoutant une étape, qui place automatiquement les messages envoyés dans la liste d'attente « push_queue ». Ainsi, lorsqu'un message est envoyé, il est enregistré dans la base de données et aussitôt ajouté à la file d'attente « push_queue » pour être traité par le service de notification installé.

Ensuite, ce dernier prend le relais. Il est chargé d'une part de maintenir une connexion avec le service APNs, et d'autre part de surveiller la table « push_queue » de la base de données du service web afin que dès qu'une entrée y est ajoutée, il la traite et la transmette à l'APNs. Une fois envoyée à l'APNs, la notification est redirigée par celui-ci vers l'appareil iOS concerné.

7.3.2 Génération du certificat

Concernant la génération du certificat pour l'application, nous avons dû nous adresser à monsieur Christophe Pignat, responsable du programme de développement Apple de la filière informatique de gestion de la HES-SO de Sierre. C'est la seule personne autorisée à effectuer cette action car Apple restreint la génération des certificats aux administrateurs respectifs de chaque programme de développement. Cette contrainte complique le processus de génération pour les développeurs mais renforce la sécurité.

8 Gestion de projet

La réalisation d'un tel projet, a nécessité la mise en place d'une planification, afin de gérer le temps mis à disposition. Le temps imparti pour ce travail était de onze semaines.

Les sujets ont été remis le seize mai et le travail était à rendre pour le seize août. Du fait que certains cours relatifs à notre formation se déroulaient encore en parallèle durant les cinq premières semaines, seulement vingt heures y étaient dédiées. Après les cours, deux semaines ont été consacrées aux examens finaux et sont à retirer du planning. Et les six semaines restantes étaient des semaines complètes, c'est-à-dire des semaines de quarante-cinq heures.

Comme nous le verrons plus bas, les heures consacrées au travail durant les premières semaines étaient souvent inférieures aux heures mises à disposition. Ceci est dû au fait que certains examens intermédiaires avaient lieux durant cette période. Cependant le temps perdu à été rattrapé dans les semaines suivant les examens finaux.

Durant la première semaine, il a fallu mettre en place une planification. Voici donc la répartition du temps sur les onze semaines à disposition, selon les tâches définies en début de projet.

Tableau 1 : Planification

	Tâche	Nb semaines *
# 01	Préparatifs et analyses	1 sem.
# 02	Formation au développement iOS	2 sem.
# 03	Mise en place du service web	1 sem.
# 04	Développement de l'application chat	2 sem.
# 05	Prise en charge des notifications	1 sem.
# 06	Finalisation du projet (Test et rédaction du rapport)	2 sem.

** Une semaine est évaluée à environ 45 heures*

Voici ensuite la répartition des tâches effectuées sur les différentes semaines, avec les heures effectivement consacrées. Les rapports hebdomadaires avec des descriptions plus détaillées des tâches se trouvent dans les annexes.

Tableau 2 : Rapport des heures

Semaine	Tâche	Heures consacrées
# 01	Définition de l'architecture	4 heures
# 02	Définition de l'architecture et planification	18 heures
# 03	Formation au développement iOS	26 heures
# 04	Formation au développement iOS	16 heures
# 05	Formation au développement iOS	16 heures
# 06	Mise en place et développement du service web	45 heures
# 07	Développement de l'application	54 heures
# 08	Développement de l'application	57.5 heures
# 09	Développement du service de notification	51 heures
# 10	Rédaction du rapport, tests et corrections	40 heures
# 11	Rédaction du rapport, tests et corrections	45 heures
	Total	372.5 heures

Compte tenu des examens intermédiaires ayant eu lieu durant les premières semaines du projet, nous pouvons constater que les horaires n'ont pas toujours pu être respectés. Cependant les heures perdues ont pu être récupérées durant la tâche de développement de l'application.

En effet nous avons mis de côté certains sujets à aborder durant la formation, sachant qu'ils n'étaient pas nécessaires pour débuter l'application. Ainsi aucun retard n'a été pris et le planning a pu être tenu pour les tâches consacrées au service web et à l'application. Par la suite, si besoin était de voir un sujet en particulier pour le développement d'une fonctionnalité, du temps y était consacré.

Nous pouvons donc conclure en disant que la planification initiale a été respectée. De plus, le projet ne comportant qu'une seule ressource, la gestion du temps était plus subjective et la planification pouvait être aisément adaptée, sans engendrer de complications.

9 Bilan

9.1 Difficultés rencontrées

Les difficultés les plus importantes rencontrées durant ce projet sont les suivantes.

9.1.1 Application Chat

Tout d'abord, n'ayant jamais eu l'occasion de développer sur la plateforme mobile d'Apple, il nous a fallu de nombreuses heures de lectures et d'exercices afin d'avoir une vue d'ensemble du développement sur iOS. Le fait de devoir ajouter un temps de formation nous a forcé à adapter le planning et limiter les fonctionnalités secondaires.

Ensuite, d'un point de vue plus technique, nous avons rencontré des difficultés au niveau de la gestion des vues sur l'application, plus précisément l'utilisation conjointe d'un contrôleur de vues par onglets (UITabBarController) et d'un contrôleur de vues par navigation (UINavigationController). Le contrôleur de vues par onglet permet la gestion générale des pages « Amis », « Discussions », « Favoris » et « Paramètres », alors que le contrôleur de vues par navigation s'occupe, quant à lui, de la gestion des pages au sein de l'onglet « Discussions » afin de permettre l'empilement des vues relatives à la discussion avec un contact sur la vue générale regroupant toutes les discussions.

Image 21 : UITabBarController

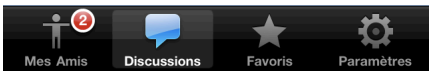


Image 22 : UINavigationController

En poursuivant sur les difficultés techniques, nous avons rencontré quelques complications avec la gestion de la mémoire de l'application. En effet, ayant principalement développé en Java, où la gestion de la mémoire est faite automatiquement, il nous a fallu du temps afin de nous habituer à cette nouveauté.

> *Par gestion de la mémoire nous faisons ici principalement référence à la libération de la mémoire. A chaque allocation de mémoire pour une instance d'objet, il est important de libérer cet espace mémoire lorsqu'il n'est plus utile. En Java, cette tâche est prise en charge par le « garbage collector », propriété de la machine virtuelle Java (JVM), qui va surveiller la mémoire et automatiquement remettre à disposition les espaces mémoires alloués qui ne sont plus utilisés.*

9.1.2 Service web

Concernant la partie service web, nous n'avons pas rencontré de difficultés majeures. Seule la prise en main de la librairie CFPropertyList, utilisée pour

sérialiser, respectivement désérialiser, les données échangées entre l'application mobile et le service web, nous a demandé un temps d'adaptation.

9.1.3 Service de notification

Au sujet du service de notification, la seule difficulté à signaler est la génération du certificat, son adaptation pour l'utilisation en PHP et finalement son intégration.

Premièrement, la génération du certificat a été compliquée. Cependant les complications engendrées étaient plutôt d'ordre géographique puisque comme nous l'avons mentionné dans le chapitre relatif au service de notification, la génération de certificats n'est autorisée qu'à la personne administrant le programme de développement. Cette personne travaillant au service informatique de la HEVs, nous avons dû nous déplacer afin de générer le certificat auprès de cette personne.

Ensuite, pour manipuler plus facilement le certificat sur le service PHP, nous avons dû adapter ce dernier en changeant son format. Une fois le certificat prêt, nous l'avons intégré au service de notification mis en place sur le serveur. Malheureusement, le tutoriel suivi pour l'adaptation du certificat comportait des erreurs, ce qui a eu pour conséquence le rejet du certificat par le service de notification. Et l'identification de ce problème nous a fait perdre beaucoup de temps.

9.2 Améliorations possibles

Les fonctionnalités définies au début de ce travail ayant été respectées, il nous est néanmoins apparu en fin de développement et en testant l'application, que certaines améliorations pouvaient y être apportées. Certaines de ces retouches devraient être effectuées avant de mettre l'application en production.

9.2.1 Application Chat

Tout d'abord, nous pourrions, comme beaucoup d'applications concurrentes, ajouter le support du partage multimédia tel que l'envoi d'images ou de sons. En effet, la majorité des applications de messagerie instantanée existantes embarquent déjà cette fonctionnalité. Et cette modification pourrait apporter un plus à l'utilisateur final, car comme on le dit souvent, une image vaut mieux que mille mots.

Ensuite, une amélioration pourrait être apportée à la page de conversation. Lors de l'écriture d'un message à un contact, le champ texte qui permet l'insertion de texte ne supporte qu'une seule ligne. Or lorsque l'utilisateur désire modifier ou contrôler le contenu de son message, la tâche peut devenir délicate si le texte est assez long. L'idéal serait donc ici de changer ce champ texte afin de permettre l'édition sur

plusieurs lignes et de le redimensionner dynamiquement en fonction du nombre de lignes.

9.2.2 Service web

Une amélioration notable serait aussi à apporter au service web. Cette modification est cette fois plus de l'ordre de l'architecture du service, de l'ordonnancement des pages et de la généralisation de l'utilisation.

Actuellement le service est composé de plusieurs pages appelées par l'application en fonction des opérations effectuées par l'utilisateur. Toutefois il serait plus approprié de centraliser les requêtes de l'application sur une seule page du service web. L'opération à effectuer serait passée en paramètre de la requête et c'est le service web qui serait ensuite chargé de faire les appels nécessaires selon la commande reçue. Cette méthode permettrait notamment de centraliser sur la page principale certaines opérations récurrentes comme la vérification de l'utilisateur, la lecture des requêtes au format « plist » et leur création. De plus, il n'y aurait plus qu'une seule URL à définir dans l'application pour contacter le service web, ce qui éviterait les erreurs.

9.3 Bilan personnel

Au niveau personnel, nous avons trouvé ce travail très intéressant. D'une part pour le nouveau langage appris, et d'autre part pour la liberté totale en matière de gestion du temps.

Il était assez audacieux au départ de se lancer dans un projet sans aucune notion du langage de développement, mais le bénéfice retiré en valait la peine. Néanmoins nous ne cachons pas qu'à certains moments du projet, la tension était à son comble, à cause du degré d'incertitude occasionné par le manque d'expérience. Mais au fil du temps l'application a commencé à prendre forme et à chaque fonctionnalité réalisée, notre assurance n'en était que renforcée.

Avec le recul, le seul point négatif à mettre en exergue est la difficulté d'organisation rencontrée en début de projet, due aux examens se déroulant en parallèle. Il aurait été plus judicieux que les examens intermédiaires aient eu lieu avant le début du travail de Bachelor, afin que les révisions n'empiètent pas sur le temps qui lui était destiné.

10 Conclusion

En définitive, nous avons pu voir comment fonctionne cette application, quelle architecture il a fallu mettre en place pour répondre aux besoins. Nous avons ensuite parcouru les différentes parties de l'architecture, afin de connaître le rôle de chacune d'entre elles.

Cette architecture est par ailleurs intéressante, car le service web qui la compose permettrait, dans un futur proche, de porter ce service sur d'autres applications développées sur des plateformes mobiles différentes.

Dans un premier temps, une version sur iPad serait la plus rapide à réaliser car une simple adaptation de l'application chat suffirait. Le seul changement étant la différence de résolution entre les écrans des deux appareils.

Par la suite il serait également imaginable de développer une version sur Android, étant donné le succès rencontré par la plateforme mobile de Google. Et enfin, d'autres adaptations ne sont pas à exclure. Le service web est indépendant et n'importe quel système peut s'y connecter du moment qu'il utilise le même format d'échange.

Toutefois, pour une utilisation plus large, le format d'échange JSON décrit plus haut serait à préférer au Plist, car il est tout de même plus rapide et plus facile à intégrer sur d'autres plateformes. Bien entendu, modifier le format d'échange implique de nombreux changements à effectuer d'une part sur le service web et d'autre part sur l'application mobile. Mais le gain en matière d'efficacité serait considérable, tant sur le plan technologique qu'au niveau du développement.

Pour terminer, conformément au cahier des charges, toutes les fonctionnalités définies en début de projet ont été respectées. Aussi bien les fonctionnalités principales que secondaires ont été développées, moyennant cependant quelques heures supplémentaires.

Pour rappel les fonctionnalités ont été séparées en deux parties distinctes : Les fonctionnalités prioritaires permettant de faire fonctionner l'application de manière basique et les fonctionnalités secondaires complétant l'application en améliorant l'expérience de l'utilisateur. Cette séparation a été faite en tenant compte de la période de formation nécessaire pour maîtriser le langage de développement sur iOS.

10.1 Mot de la fin

En conclusion, ce travail a été pour nous l'occasion de nous impliquer totalement dans l'élaboration d'un projet du début à la fin. Ceci était relativement nouveau, puisque quasiment tous les autres projets effectués durant notre formation étaient des travaux de groupes. Mais le fait de pouvoir gérer totalement le temps à disposition pour ce projet a été un facteur motivant.

De plus, l'initiative de choisir un langage de développement totalement inconnu pour ce travail de Bachelor a été grandement enrichissante et nous a permis d'étoffer nos connaissances.

11 Remerciements

Nous tenons à remercier cordialement toutes les personnes ayant contribué, de quelque manière que ce soit, à la réalisation de ce travail de Bachelor.

Nous souhaitons tout particulièrement remercier :

Mme Nicole Glassey, notre professeur responsable, de l'encadrement qu'elle nous a fourni et de son implication tout au long de ce travail.

M. Guillaume Fort, assistant Cyberlearn à l'institut d'Informatique de Gestion de la HES-SO Valais, de son aide précieuse concernant tous les aspects techniques du projet, notamment la communication entre l'application iPhone et le service web, mais également d'avoir répondu à nos questions lorsque nous rencontrions des difficultés.

M. Christophe Pignat, responsable du programme pour les développeurs iOS de l'institut d'Informatique de Gestion, de nous avoir permis de générer un certificat pour notre application.

Merci aussi aux personnes nous ayant aidé à la relecture de ce rapport, pour le temps qu'ils y ont consacré.

Enfin, un grand merci à tout notre entourage pour leurs encouragements et leur soutien tout au long de notre formation.

12 Déclaration sur l'honneur

Je déclare, par ce document, que j'ai effectué le travail de diplôme ci-annexé seul, sans autre aide que celles dûment signalées dans les références, et que je n'ai utilisé que les sources expressément mentionnées. Je ne donnerai aucune copie de ce rapport à un tiers sans l'autorisation conjointe du RF et du professeur chargé du suivi du travail de diplôme, y compris au partenaire de recherche appliquée avec lequel j'ai collaboré, à l'exception des personnes qui m'ont fourni les principales informations nécessaires à la rédaction de ce travail et que je cite ci-après :

- Mme Nicole Glassey, professeur en charge du projet
- M. Guillaume Fort, assistant Cyberlearn à la HES-SO Valais

Lieu, date et signature de Daniel Hunacek

13 Sources

13.1 Bibliographie

Beginning iPhone 3 Development: Exploring the iPhone SDK

Jeff LaMarche, Dave Mark

APress (21 juillet 2011)

Beginning iPhone 4 Development: Exploring the iOS SDK

Jack Nutting, Jeff LaMarche, Dave Mark

APress (1er février 2011)

13.2 Webographie

13.2.1 Développement iPhone

- http://www.techotopia.com/index.php/An_iOS_4_iPad_Core_Data_Turorial_(Xcode_4)
  Dernière visite : 11.07.2011
- http://developer.apple.com/library/ios/#documentation/*
  Dernière visite : 11.07.2011
- http://www.raywenderlich.com/934/core-data-tutorial-getting-started
  Dernière visite : 12.07.2011
- http://www.ipup.fr/news.php?id=1859
  Dernière visite : 12.08.2011
- http://www.ipup.fr/news.php?id=116
  Dernière visite : 12.08.2011
- http://ipup.fr/forum/viewtopic.php?id=130
  Dernière visite : 12.08.2011
- http://ipgames.wordpress.com/tutorials/writeread-data-to-plist-file/
  Dernière visite : 12.08.2011
- http://www.iostipsandtricks.com/using-different-uitableviewcell-styles/
  Dernière visite : 12.08.2011
- http://blog.tolik.org/2010/04/iphone-sdk-sms-balloon-cell.html
  Dernière visite : 12.08.2011
- https://github.com/hboon/GrowingTextView/
  Dernière visite : 12.08.2011

- http://samsoff.es/posts/web-services-with-cocoa-surprise
  Dernière visite : 12.08.2011
- http://stackoverflow.com/ (*forums*)
  Dernière visite : -

13.2.2 Architecture

- http://www.tutomobile.fr/model-view-controller/17/10/2010/
  Dernière visite : 12.08.2011
- http://fr.wikipedia.org/wiki/Modèle-Vue-Contrôleur
  Dernière visite : 10.08.2011
- http://developer.apple.com/technologies/ios/data-management.html
  Dernière visite : 12.08.2011

13.2.3 Service web

- https://github.com/rodneyrehm/CFPropertyList
  Dernière visite : 12.08.2011
- http://stackoverflow.com/ (*forums*)
  Dernière visite : -

13.2.4 Service de notification Push

- http://www.raywenderlich.com/3443/apple-push-notifrication-services-tutorial-part-12
  Dernière visite : 25.07.2011
- http://www.raywenderlich.com/3525/apple-push-notification-services-tutorial-part-2
  Dernière visite : 04.08.2011
- http://developer.apple.com/library/ios/documentation/NetworkingInternet/Conceptual/RemoteNotificationsPG/RemoteNotificationsPG.pdf
  Dernière visite : 12.08.2011
- http://www.easyapns.com/
  Dernière visite : 12.08.2011
- http://stackoverflow.com/ (*forums*)
  Dernière visite : -

14 Glossaire

D

DMZ	Demilitarized Zone (sous-réseau isolé)

F

Framework	Kit de composants logiciels structurels

I

iOS	Plateforme mobile d'Apple

J

JSON	JavaScript Object Notation Format de données textuel, générique

M

MMHS	My Mobile Health Space
MVC	Modèle – Vue – Contrôleur
MySQL	Système de gestion de base de données

P

PHP	General-purpose scripting language
Plist	Property List Fichier utilisé pour sérializer des objets

R

Ra&D	Recherche appliquée et développement

S

SHA1	Secure Hash Algorithm, fonction de hachage cryptographique
SDK	Software Development Kit Outil pour développer des applications d'un type défini

U

UDID	Identifiant unique d'un appareil Apple
URL	Uniform Resource Locator

X

XML	Extensible Markup Language Langage informatique de balisage générique

15 Table des illustrations

15.1 Images

Image 1 : Schéma d'architecture 5
Image 2 : Ecran de login 8
Image 3 : Ecran "Mes Amis" 8
Image 4 : Ecran d'édition de "Mes Amis" 9
Image 5 : Ecran de recherche d'amis 9
Image 6 : Ecran "Discussions" 10
Image 7 : Mode d'édition de "Discussions" 10
Image 8 : Ecran de conversation 11
Image 9 : Menu contextuel d'une conversation 11
Image 10 : Ecran "Favoris" 12
Image 11 : Mode d'édition de "Favoris" 12
Image 12 : Ecran "Paramètres" 13
Image 13 : Schéma MVC 14
Image 14 : Entités dans l'application 15
Image 15 : « Managed Objects » dans l'application 16
Image 16 : Schéma de la base de données 21
Image 17 : Autoriser les notifications Push 24
Image 18 : Plateforme des développeurs Apple 25
Image 19 : Une notification 26
Image 20 : Exemple d'un badge 27
Image 21 : UITabBarController 31
Image 22 : UINavigationController 31

15.2 Tableaux

Tableau 1 : Planification 29

Tableau 2 : Rapport des heures 30

15.3 Exemple de code

Code 1 : Exemple de lecture d'un fichier "Plist" en PHP 18

Code 2 : Exemple de payload 26

16 Annexes

Liste des annexes :

- Cahier des charges

Tous les autres documents relatifs au projet (rapports hebdomadaires, écrans, code source, etc.) sont dans le cd en annexe.

Daniel Hunacek TB – Application chat sur iOS 31.05.11

Cahier des charges

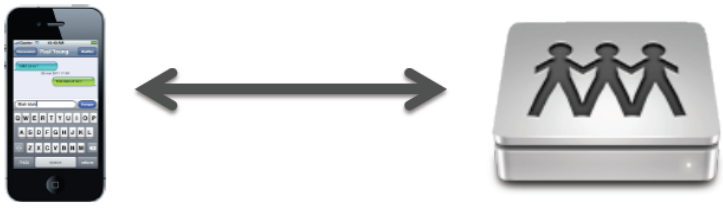
Introduction

Ce projet a pour but de développer une application de chat pouvant ensuite être intégrée dans une application plus conséquente visant à offrir un environnement personnalisé pour les patients sous traitement. Cette application faisant office de conteneur est appelée My Mobile Health Space (MMHS) et sera ciblée sur la prévention, la prise en charge personnelle du patient et son implication dans le processus thérapeutique.

Cette application de chat est destinée au système iOS, utilisé par les iPhone, iPod touch et iPad. Elle sera tout d'abord créée en tant qu'application à part entière puis intégrée dans l'application MMHS par la suite afin d'offrir aux patients la possibilités de parler entre eux de leurs problèmes ou astuces, mais également de garder contact avec des personnes dans la même situation.

Description générale

Ce projet suivra une architecture client – serveur et sera donc subdivisé en deux parties. La première sera le client, c'est-à-dire l'application de chat, et l'autre sera le serveur, qui gérera les échanges de messages.



Client Serveur

Technologies utilisées

L'application chat sera développée en Objective-C, langage propre au développement sur iOS quant au serveur, il sera développé en php avec une base de données en MySQL. Le langage d'échange entre le serveur et le client sera en JSON.

Daniel Hunacek TB – Application chat sur iOS 31.05.11

Utilisateurs visés

Les utilisateurs concernés sont, dans un premier temps des jeunes personnes atteintes d'asthme, puis par la suite et dans une optique plus globale, toute personne sous traitement.

Leur but est de pouvoir dialoguer avec des personnes dans le même cas, qui pourront leur donner des conseils et avec qui ils pourront garder contact.

Fonctionnalités

L'application offrira plusieurs fonctionnalités :

1. Une application privée

   Cette application est destinée uniquement à des personnes bien précises. C'est pourquoi il faudra s'authentifier au démarrage de l'application à l'aide d'un identifiant et d'un mot de passe. Ces derniers seront fournis par une personne responsable de l'administration, qui les ajoutera à partir d'une interface simple directement sur le serveur (phpMyAdmin).

2. Un chat

   L'utilisateur pourra, une fois connecté, ajouter des amis dans sa liste d'amis. Il pourra rechercher des amis par prénom, nom et identifiant et leur faire une demande d'ajout. La demande d'amitié devra ensuite être validée par l'utilisateur concerné pour que le lien d'amitié soit créé.

   Il sera ensuite possible de dialoguer avec les contacts de sa liste d'amis. Les messages seront envoyés via un serveur distant réceptionnant les messages et les acheminant au destinataire.

   Chaque fois que l'utilisateur lancera l'application, cette dernière synchronisera la liste des messages avec le serveur. Ainsi tout nouveau message sera immédiatement visible. Cette même synchronisation se fera ensuite de manière automatique à intervalle régulier lors de l'utilisation.

   Il sera possible également de modifier certains paramètres relatifs à l'application dans la rubrique « Paramètres ». Ces options seront définies selon les besoins de l'application.

3. Des trucs et astuces

   Ce chat servant principalement à échanger des conseils entre utilisateurs, il permettra de mettre en évidence certains des messages reçus qui seraient pertinents afin de les conserver en tant que trucs et astuces. Ces messages seront ensuite disponibles via la rubrique « Favoris » afin de les retrouver rapidement.

4. Toujours connecté

   Afin de savoir lorsque de nouveaux messages seront disponibles, l'application utilisera l'API de notification push proposée par Apple sur son système iOS, dans le but d'avertir l'utilisateur de la venue d'un nouvel événement (demande d'ajout d'ami, nouveau message).

Daniel Hunacek TB – Application chat sur iOS 31.05.11

Liste des fonctionnalités avec priorités

Fonctionnalités prioritaires

- Page de login permettant de s'identifier auprès du serveur.
- Pages « mes amis » avec possibilité d'ajouter et supprimer des amis de sa liste.
- Fonction de recherche d'amis par nom, prénom et identifiant.
- Fonction « chat » :
  - Page « discussions » avec liste des conversations et possibilité de créer ou supprimer une conversation.
  - Envoi de messages à un contact.
  - Réception de messages

Fonctionnalités secondaires

- Possibilité de recevoir des notifications PUSH.
- Possibilité de mettre en favori un message reçu.
- Page « favoris » regroupant les favoris avec possibilité de les supprimer.
- Page « paramètres » comprenant toutes les options relatives à l'application.

Prototypes écran

Les prototypes écran sont en annexes dans le dossier « prototypes_ecran ».

Planning

Le planning du projet se trouve en annexe dans le document « planification.docx ». Une version plus détaillée du planning de projet se trouve dans le document « TB_Planification_initiale.mpp ».